AprilTag: A robust and flexible visual fiducial system

Edwin Olson

University of Michigan

ebolson@umich.edu

http://april.eecs.umich.edu

Abstract— While the use of naturally-occurring features is a central focus of machine perception, artificial features (fiducials) play an important role in creating controllable experiments, ground truthing, and in simplifying the development of systems where perception is not the central objective.

We describe a new visual fiducial system that uses a 2D bar code style "tag", allowing full 6 DOF localization of features from a single image. Our system improves upon previous systems, incorporating a fast and robust line detection system, a stronger digital coding system, and greater robustness to occlusion, warping, and lens distortion. While similar in concept to the ARTag system, our method is fully open and the algorithms are documented in detail.

I. INTRODUCTION

Visual fiducials are artificial landmarks designed to be easy to recognize and distinguish from one another. Although related to other 2D barcode systems such as QR codes [1], they have significantly goals and applications. With a QR code, a human is typically involved in aligning the camera with the tag and photographs it at fairly high resolution obtaining hundreds of bytes, such as a web address. In contrast, a visual fiducial has a small information payload (perhaps 12 bits), but is designed to be automatically detected and localized even when it is at very low resolution, unevenly lit, oddly rotated, or tucked away in the corner of an otherwise cluttered image. Aiding their detection at long ranges, visual fiducials are comprised of many fewer data cells: the alignment markers of a QR tag comprise about 268 pixels (not including required headers or the payload), whereas the visual fiducials described in this paper range from about 49 to 100 pixels, *including* the payload.

Unlike 2D barcode systems in which the position of the barcode in the image is unimportant, visual fiducial systems provide camera-relative position and orientation of a tag. Fiducial systems also are designed to detect multiple markers in a single image.

Visual fiducial systems are perhaps best known for their application to augmented reality, which spurred the development of several popular systems including ARToolkit [2] and ARTag [3]. Real-world objects can be augmented with visual fiducials, allowing virtually-generated imagery to be super-imposed. Similarly, visual fiducials can be used for basic motion capture [4].

Visual fiducial systems have been used to improve human/robot interaction, allowing humans to signal commands (such as "follow me" or "wait here") by flashing an appropriate card to a robot [5]. Planar tags have also been used to
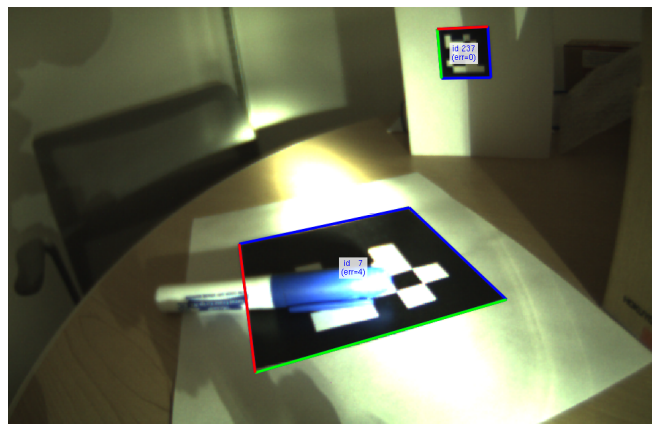

Fig. 1. Example detections. This paper describes a visual fiducial system based on 2D planar targets. The detector is robust to lighting variation and occlusions and produces accurate localizations of the tags.

generate user interfaces that overlay robots' plans and task assignments onto a head-mounted display [6].

Performance evaluation and benchmarking of robot systems have become central issues for the research community; visual fiducials are particularly helpful in this domain. For example, fiducials can be used to generate ground-truth robot trajectories and close control loops [7]. Similarly, artificial features can make it possible to evaluate Simultaneous Localization and Mapping (SLAM) algorithms under controlled algorithms [8]. Robotics applications have led to the development of additional tag detection systems [9], [10].

Designing robust fiducials while minimizing the size required is a challenging both from a marker detection standpoint (which pixels in the image correspond to a tag?) and from a error-tolerant data coding standpoint (which tag is it?)

In this paper, we describe a new visual fiducial system that significantly improves performance over previous systems. The central contributions of this paper are:

- We describe a method for robustly detecting visual fiducials. We propose a graph-based image segmentation algorithm based on local gradients that allows lines to be precisely estimated. We also describe a quad extraction method that can handle significant occlusions.
- We demonstrate that our detection system provides significantly better localization accuracy than previous systems.
- We describe a new coding system that address problems

unique to 2D barcoding systems: robustness to rotation, and robustness to false positives arising from natural imagery. As demonstrated by our experimental results, our coding system provides significant theoretical and real-world benefits over previous work.

- We specify and provide results on a set of benchmarks which will allow better comparisons of fiducial systems in the future.

In contrast to previous methods (including ARTag and Studierstube Tracker), our implementation is released under an Open Source license, and its algorithms and implementation are well documented. The closed nature of these other systems was a challenge for our experimental evaluation. For the comparisons in this paper, we have used the limited publicly-available information to enable as many objective comparisons as possible. On the other hand, ARToolkitPlus is open source and so we were able to make a more detailed comparison. In addition to code, we are also making our evaluation code available in order to make it easier for future authors to perform comparisons.

In the next section, we review related work. We describe our method in the following two sections: the tag detector in Section 3, and the coding system in Section 4. In Section 5, we provide an experimental evaluation of our methods, comparing them to previous algorithms.

II. Previous Work

ARToolkit [11] was among the first tag tracking systems, and was targeted at artificial reality applications. Like the systems that would follow, its tags contained a square-shaped payload surrounded by a black border. It differed, however, in that its payload was not directly encoded in binary: instead, it used symbols such as the latin character 'A'. When decoding a tag, the payload of the tag (sampled at high resolution) was correlated against a database of known tags, with the best-correlating tag reported to the user. A major disadvantage of this approach is the computational cost associated with decoding tags, since each template required a separate, slow correlation operation. A second disadvantage is that it is difficult to generate templates that are approximately orthogonal to each other.

The tag detection scheme used by ARToolkit is based on a simple binarization of the input image based on a user-specified threshold. This scheme is very fast, but not robust to changes in illumination. In general, ARToolkit's detections can not handle even modest occlusions of the tag's border.

ARTag [3] provided improved detection and coding schemes. Like our own approach, the detection mechanism was based on the image gradient, making it robust to changes in lighting. While the details of the detector algorithm are not public, ARTag's detection mechanism is able to detect tags whose border is partially occluded. ARTag also provided the first coding system based on forward error correction, which made tags easier to generate, faster to correlate, and provided greater orthogonality between tags.

The performance of ARTag inspired several improvements to ARToolkit, which evolved into ARToolkitPlus [2], and

finally Studierstube Tracker [12]. These versions introduced digitally-encoded payloads like those used in ARTag. Despite being later work, our experiments show that these encoding systems do not perform as well as that used by ARTag, which in turn is outperformed by our coding system.

In addition to monochrome tags, other coding systems have been developed. For example, color information has been used to increase the amount of information that can be encoded [13], [14]. Tags using retro-reflectors [15] have also been used. A particularly interesting approach is that used by Bokode [16], which exploits the bokeh effect to detect extremely small tags by intentionally defocusing the camera.
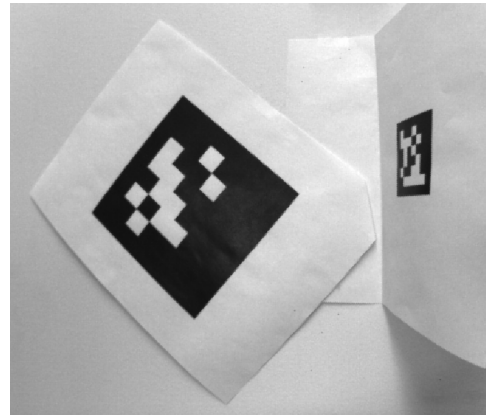

Fig. 2. Input image. This paper describe the processing of this sample image which contains two tags. This example is purposefully simple for explanatory reasons, though note that the tags are not rigidly planar. See Fig. 1 for a more challenging result.

Besides two-dimensional barcodes, a number of other artificial landmarks have been developed. Overhead cameras have been used to track robots equipped with blinking LEDs [17]. In contrast, the NorthStar system puts the visual fiducials on the ceiling [18]. Two-dimensional planar systems, like the one described in this paper, have two main advantages over LED-based systems: the targets can be cheaply printed on a standard printer, and they provide 6DOF position estimation without the need for multiple LEDs.

III. Detector

Our system is composed of two major components: the tag detector and the coding system. In this section, we describe the detector whose job is to estimate the position of possible tags in an image. Loosely speaking, the detector attempts to find four-sided regions ("quads") that have a darker interior than their exterior. The tags themselves have black and white borders in order to facilitate this (see Fig. 2).

The detection process is comprised of several distinct phases, which are described in the following subsections and illustrated using the example shown in Fig. 2.

Note that the quad detector is designed to have a very low false negative rate, and consequently has a high false positive rate. We rely on the coding system (described in the next section) to reduce this false positive rate to useful levels.
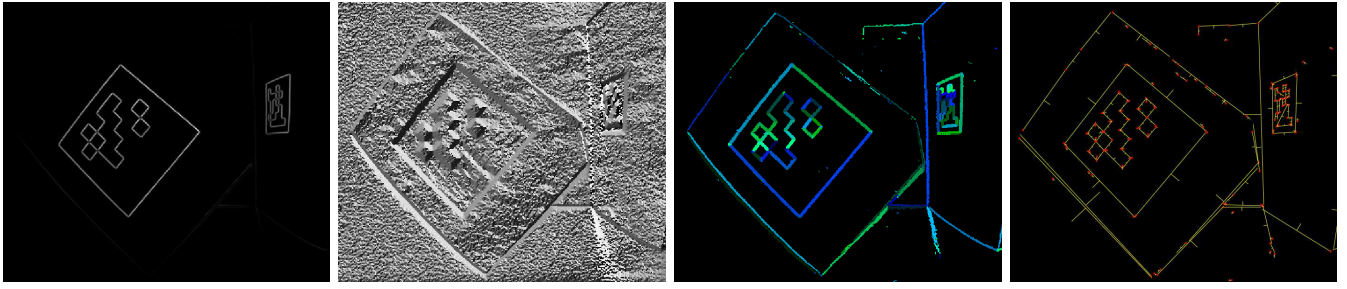
Fig. 3. Early processing steps. The tag detection algorithm begins by computing the gradient at every pixel, computing their magnitudes (first) and direction (second). Using a graph-based method, pixels with similar gradient directions and magnitude are clustered into components (third). Using weighted least squares, a line segment is then fit to the pixels in each component (fourth). The direction of the line segment is determined by the gradient direction, so that segments are dark on the left, light on the right. The direction of the lines are visualized by short perpendicular "notches" at their midpoint; note that these "notches" always point towards the lighter region.

A. Detecting line segments

Our approach begins by detecting lines in the image. Our approach, similar in basic approach to the ARTag detector, computes the gradient direction and magnitude at every pixel (see Fig. 3) and agglomeratively clusters the pixels into components with similar gradient directions and magnitudes.

The clustering algorithm is similar to the graph-based method of Felzenszwalb [19]: a graph is created in which each node represents a pixel. Edges are added between adjacent pixels with an edge weight equal to the pixels' difference in gradient direction. These edges are then sorted and processed in terms of increasing edge weight: for each edge, we test whether the connected components that the pixels belong to should be joined together. Given a component n, we denote the range of gradient directions as $D(n)$ and the range of magnitudes as $M(n)$. Put another way, $D(n)$ and $M(n)$ are scalar values representing the difference between the maximum and minimum values of the gradient direction and magnitude respectively. In the case of $D()$, some care must be taken to handle 2π wrap-around. However, since useful edges will have a span of much less than π degrees, this is straightforward. Given two components n and m, we join them together if both of the conditions below are satisfied:

$$
\begin{aligned}
D(n \cup m) &\leq \min(D(n), D(m)) + K_D/|n \cup m| \quad (1)\\
M(n \cup m) &\leq \min(M(n), M(m)) + K_M/|n \cup m|
\end{aligned}
$$

The conditions are adapted from [19] and can be intuitively understood: small values of $D()$ and $M()$ indicate components with little intra-component variation. Two clusters are joined together if their union is about as uniform as the clusters taken individually. A modest increase in intra-component variation is permitted via the K_D and K_M parameters, however this rapidly shrinks as the components become larger. During early iterations, the K parameters essentially allow each component to "learn" its intra-cluster variation. In our experiments, we used $K_D = 100$ and $K_M = 1200$, though the algorithm works well over a broad range of values.

For performance reasons, the edge weights are quantized and stored as fixed-point numbers. This allows the edges to
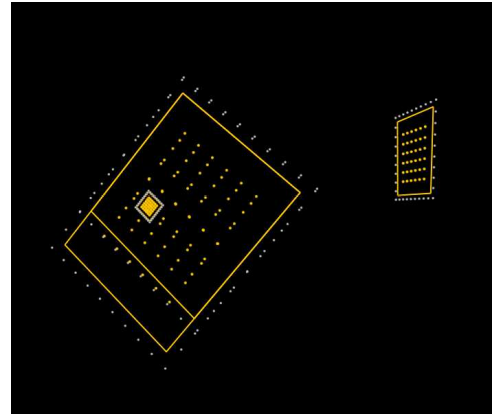

Fig. 4. Quad detection and sampling. Four quads are detected in the image (which contains two tags). The third detected quad corresponds to three of the edges of the foreground tag plus the edge of the paper (See Fig. 2). A fourth quad is detected around one of the payload bits of the larger tag. These two extraneous detections are eventually discarded because their payload is invalid. The white dots correspond to samples around the tags border which are used to fit a linear model of intensity of "white" pixels; a model is similarly fit for the black pixels. These two models are used to threshold the data payload bits, shown as yellow dots.

be sorted using a linear-time counting sort [20]. The actual merging operation can be efficiently carried out by the union-find algorithm [20] with the upper and lower bounds of gradient direction and magnitude stored in a simple array indexed by each component's representative member.

This gradient-based clustering method is sensitive to noise in the image: even modest amounts of noise will cause local gradient directions to vary, inhibiting the growth of the components. The solution to this problem is to low-pass filter the image [19], [21]. Unlike other problem domains where this filtering can blur useful information in the image, the edges of a tag are intrinsically large-scale features (particularly in comparison to the data field), and so this filtering does not cause information loss. We recommend a value of $\sigma = 0.8$.

Once the clustering operation is complete, line segments are fit to each connected component using a traditional least-squares procedure, weighting each point by its gradient magnitude (see Fig. 3). We adjust each line segment so that the dark side of the line is on its left, and the light side is

on its right. In the next phase of processing, this allows us to enforce a winding rule around each quad.

The segmentation algorithm is the slowest phase in our detection scheme. As an option, this segmentation can be performed at half the image resolution with a 4x improvement in speed. The sub-sampling operation can be efficiently combined with the recommended low-pass filter. The consequence of this optimization is a modestly reduced detection range, since very small quads may no longer be detected.

B. Quad detection

At this point, a set of directed line segments have been computed for an image. The next task is to find sequences of line segments that form a 4-sided shape, i.e., a quad. The challenge is to do this while being as robust as possible to occlusions and noise in the line segmentations.

Our approach is based on a recursive depth-first search with a depth of four: each level of the search tree adds an edge to the quad. At depth one, we consider all line segments. At depths two through four, we consider all of the line segments that begin "close enough" to where the previous line segment ended *and* which obey a counter-clockwise winding order. Robustness to occlusions and segmentation errors is handled by adjusting the "close enough" threshold: by making the threshold large, significant gaps around the edges can be handled. Our threshold for "close enough" is twice the length of the line plus five additional pixels. This is a large threshold which leads to a low false negative rate, but also results in a high false positive rate.

We populate a two-dimensional lookup table to accelerate queries for line segments that begin near a point in space. With this optimization, along with early rejection of candidate quads that do not obey the winding rule, or which use a segment more than once, the quad detection algorithm represents a small fraction of the total computational requirements.

Once four lines have been found, a candidate quad detection is created. The corners of this quad are the intersections of the lines that comprise it. Because the lines are fit using data from many pixels, these corner estimates are accurate to a small fraction of a pixel.

C. Homography and extrinsics estimation

We compute the 3×3 homography matrix that projects 2D points in homogeneous coordinates from the tag's coordinate system (in which $[0\ 0\ 1]^T$ is at the center of the tag and the tag extends one unit in the $\hat{x}$ and $\hat{y}$ directions) to the 2D image coordinate system. The homography is computed using the Direct Linear Transform (DLT) algorithm [22]. Note that since the homography projects points in homogeneous coordinates, it is defined only up to scale.

Computation of the tag's position and orientation requires additional information: the camera's focal length and the physical size of the tag. The 3×3 homography matrix (computed by the DLT) can be written as the product of the 3×4 camera projection matrix P (which we assume is known) and the 4×3 truncated extrinsics matrix E. Extrinsics

matrix are typically 4×4, but every position on the tag is at $z = 0$ in the tag's coordinate system. Thus, we can rewrite every tag coordinate as a 2D homogeneous point with z implicitly zero, and remove the third column of the extrinsics matrix, forming the truncated extrinsics matrix. We represent the rotation components of P as R_{ij} and the translation components as T_k. We also represent the unknown scale factor as s:

$$\begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} = sPE$$

$$= s \begin{bmatrix} 1/f_x & 0 & 0 & 0 \\ 0 & 1/f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{00} & R_{01} & T_x \\ R_{10} & R_{11} & T_y \\ R_{20} & R_{21} & T_z \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Note that we cannot directly solve for E because P is rank deficient. We can expand the right hand side of Eqn. 2, and write the expression for each h_{ij} as a set of simultaneous equations:

$$\begin{align} h_{00} &= sR_{00}/f_x \quad (3)\\ h_{01} &= sR_{01}/f_x \\ h_{02} &= sT_x/f_x \\ &\ldots \end{align}$$

These are all easily solved for the elements of R_{ij} and T_k except for the unknown scale factor s. However, since the columns of a rotation matrix must all be of unit magnitude, we can constrain the magnitude of s. We have two columns of the rotation matrix, so we compute s as the geometric the geometric average of their magnitudes. The sign of s can be recovered by requiring that the tag appear in front of the camera, i.e., that $T_z < 0$. The third column of the rotation matrix can be recovered by computing the cross product of the two known columns, since the columns of a rotation matrix must be orthonormal.

The DLT procedure and the normalization procedure above do not guarantee that the rotation matrix is strictly orthonormal. To correct this, we compute the polar decomposition of R, which yields a proper rotation matrix while minimizing the Frobenius matrix norm of the error [23].

IV. PAYLOAD DECODING

The final task is to read the bits from the payload field. We do this by computing the tag-relative coordinates of each bit field, transforming them into image coordinates using the homography, and then thresholding the resulting pixels. In order to be robust to lighting (which can vary not only from tag to tag, but also within a tag), we use a spatially-varying threshold.

Specifically, we build spatially-varying model of the intensity of "black" pixels, and a second model for the intensity of "white" models. We use the border of the tag, which contains known examples of both white and black pixels, to learn this model (see Fig. 4). We use the following intensity model:

$$I(x, y) = Ax + Bxy + Cy + D \quad (4)$$

This model has four parameters which are easily computed using least squares regression. We build two such models, one for black, the other for white. The threshold used when decoding data bits is then just the average of the predicted intensity values of the black and white models.

V. CODING SYSTEM

Once the data payload is decoded from a quad, it is the job of the coding system to determine it is valid or not. The goals of a coding system are to:

- Maximize the number of distinguishable codes
- Maximize the number of bit errors that can be detected or corrected
- Minimize the false positive/inter-tag confusion rate
- Minimize the total number of bits per tag (and thus the size of the tag)

These goals are often in conflict, and so a given code represents a trade-off. In this section, we describe a new coding system based on lexicodes that provides significant advantages over previous methods. Our procedure can generate lexicodes with a variety of properties, allowing the user to use a code that best fits their needs.

A. Methodology

We propose the use of a modified lexicode [24]. Classical lexicodes are parameterized by two quantities: the number of bits n in each codeword and the minimum Hamming distance between any two codewords d. Lexicodes can correct $\lfloor (d-1)/2 \rfloor$ bit errors and detect $d/2$ bit errors. For convenience, we will denote a 36 bit encoding with a minimum Hamming distance of 10 (for example) as a 36h10 code.

Lexicodes derive their name from the heuristic used to generate valid codewords: candidate codewords are considered in lexicographic order (from smallest to largest), adding new codewords to the codebook when they are at least a distance d from every codeword previously added to the codebook. While very simple, this scheme is often very close to optimal [25].

In the case of visual fiducials, the coding scheme must be robust to rotation. In other words, it is critical that when a tag is rotated by 90, 180, or 270 degrees, that it still have a Hamming distance of d from every other code. The standard lexicode generation algorithm does not guarantee this property. However, the standard generation algorithm can be trivially extended to support this: when testing a new candidate codeword, we can simply ensure that all four rotations have the required minimum Hamming distance. The fact that the lexicode algorithm can be easily extended to incorporate additional constraints is an advantage of our approach.

Some codewords, despite satisfying the Hamming distance constraint, are poor choices. For example, a code word consisting of all zeros would result in a tag that looks like a single black square. Such simple geometric patterns commonly occur in natural scenes, resulting in false positives.

The ARTag encoding system, for example, explicitly forbids two codes because they are too likely to occur by chance.

Rather than identify problematic tags manually, we further modify the lexicode generation algorithm by rejecting candidate codewords that result in simple geometric patterns. Our metric is based on the number of rectangles required to generate the tag's 2D pattern. For example, a solid pattern requires just one rectangle, while a black-white-black stripe would require two rectangles (one large black rectangle with a smaller white rectangle drawn second). Our hypothesis, supported by experimental results later in this paper, is that tag patterns with high complexity (which require many rectangles to be reconstructed) occur less frequently in nature and thus lead to lower false positive rates.

Using this idea, we again modify the lexicode generation algorithm to reject candidate codewords that are too simple. We approximate the number of rectangles required to generate the tag's pattern with a simple greedy approach that repeatedly considers all possible rectangles and adds the one that reduces the error the most. Since the tags are generally very small, this computation is not a bottleneck. Tags with a minimum complexity less than a threshold (typically 10 in our experiments) are rejected. The appropriateness and effectiveness of this heuristic are demonstrated in the results section.

Lastly, we have empirically observed lower false positive scores by making one more modification to the lexicode generation algorithm. Rather than test codewords in order (0, 1, 2, 3, ...), we consider (b, b+1p, b+2p, b+3p, ...) where b is an arbitrary number, p is a large prime, and the lowest n bits are kept at each step. Intuitively, the tags generated by this method have greater entropy at every bit position; the lexicographic order, on the other hand, favors small-valued codes. The disadvantage of this method is that fewer distinguishable codes are created: the lexicographic ordering tends to pack codewords quite densely, whereas the more random order results in a less efficient packing of codewords.

To summarize, we use a lexicode system that can generate codes for any arbitrary tag size (e.g., 3x3, 4x4, 5x5, 6x6) and minimum Hamming distance. Our approach explicitly guarantees the minimum Hamming distance for all four rotations of each tag and eliminates tags which are of low geometric complexity. Computing the tags can be an expensive operation, but is done offline. Small tags (5x5) can be easily computed in seconds or minutes, but larger tags (6x6) can take several days of CPU time. Many useful code families are already computed and distributed with our software; most users will not need to generate their own code families.

B. Error correction analysis

Theoretical false positive rates can be easily estimated. Assume that a false quad is identified and that the bit pattern is random. The probability of a false positive is the fraction of codewords which are accepted as valid tags versus the total number of possible codewords, 2^n. More aggressive error correction increases this rate, since it increases the number

of codewords that are accepted. This unavoidable increase in error rate is illustrated for the 36h10 and 36h15 codes below:

Bits corrected	36h10 FP (%)	36h15 FP (%)
0	0.000001	0.000000
1	0.000041	0.000002
2	0.000744	0.000029
3	0.008714	0.000341
4	0.074459	0.002912
5	0.495232	0.019370
6	N/A	0.104403
7	N/A	0.468827

Of course, the better performance of the 36h15 encoding comes at a price: there are only 27 distinguishable codewords, as opposed to 36h10's 2221 distinguishable codewords.

Our coding scheme is significantly stronger than previous schemes, including that used by ARTag and both systems used by ARToolkitPlus: our coding system achieves a greater minimum Hamming distance between all pairs of codewords while encoding a larger number of distinguishable ids. This improvement in minimum Hamming distance is illustrated in Fig. 5 and in the table below:

Encoding Scheme	Length	Unique codes	Min. Hamming
ARToolkit+ (simple)	36	512	4
ARToolkit+ (BCH)	36	4096	2
ARTag	36	2046	4
Proposed (36h9)	36	4164	9
Proposed (36h10)	36	2221	10

In order to decode a possibly-corrupted code word, the Hamming distance between the code word and each valid code word in the code book is computed. If the best match has a Hamming distance less than the user-specified threshold, a detection is reported. By specifying this threshold, the user is able to control the tradeoff between false positives and false negatives.

A disadvantage of our method is that this decoding process takes linear time in the size of the codebook, since every valid codeword must be considered. However, the coefficient is so small that the computational complexity is negligible in comparison to the other image processing steps.

For a given coding scheme, larger tags (i.e., those with 36 bits versus 25 bits) have dramatically better coding performance than smaller tags, although this comes at a price. All other things being equal, the range at which a given camera can read a 36 bit tag will be shorter than the range at which the same camera can read a 16 or 25 bit tag. However, the benefit in range for smaller tags is quite modest due to the 4 pixel overhead of the borders; only a 25% improvement in detection range can be expected by using 16 bit tags instead of 36 bit tags. Thus, it is only in the most range-sensitive applications where smaller tags are advantageous.

VI. EXPERIMENTAL RESULTS

A. Empirical Experiments

A key question we wish to answer is whether our analytical predictions regarding false positive rates holds in real-world imagery. To answer this question, we used a standard
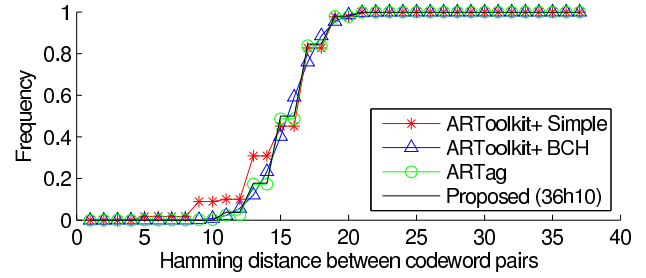

Fig. 5. Hamming distances. Good codes have large Hamming distances between valid codewords. Shown above are the Hamming distances for several contemporary systems. Note that our coding scheme *guarantees* a minimum Hamming distance by construction, whereas other systems have some very similar codewords which leads to higher inter-tag confusion rates.

image corpus from the LabelMe dataset [26], which contains 180,829 images from a wide variety of indoor and outdoor environments. Since none of these images contain one of our tags, we can measure the false positive rate of our coding systems by using these images.
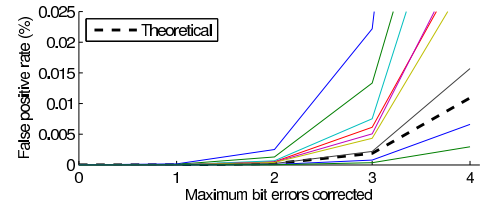

Fig. 6. Empirical false positives versus tag complexity. Our theoretical error rates assume that all codewords are equally likely to occur by chance in a real-world environment. Our hypothesis is that real-world environments are biased towards codes which have a low rectangle covering complexity and that by selecting codewords that have high rectangle covering complexity, we can decrease false positive rates. This hypothesis is validated by the graph above, which shows empirical false positive rates from the LabelMe dataset (solid lines) for rectangle covering complexities from c=2 to c=10. At complexities of c=9 and c=10, the false positive rate drops below the rate predicted by the pessimistic model that real-world payloads are randomly distributed.

Evaluation of complexity heuristic: We first wish to evaluate our hypothesis that the false positive rate can be reduced by imposing our geometric complexity heuristic to reject candidate codewords. To do this, we generated ten variants of the 25h9 family with minimum complexities ranging from 1 to 10. In Fig. 6, the false positive rate is given for each of these complexities as a function of the maximum number of bit errors corrected. Also displayed is the theoretical false positive rate which was based on the assumption that data payloads are randomly distributed.

Fig. 6 clearly demonstrates that our heuristic is effective in reducing the false positive rate. Interestingly, once the complexity exceeds 8, the performance is actually *better* than predicted by theory.

Comparison to other coding schemes: We next compare the false positive of our rate of our coding systems to those used by ARToolkitPlus and ARTag.

Using the same real-world imagery dataset, we plot the empirical false positive rates for five codes in Fig. 7.
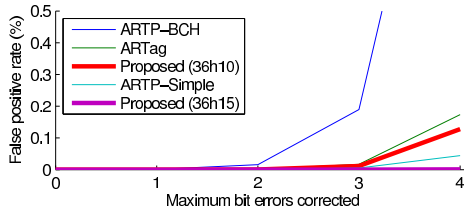
Fig. 7. Empirical false positives. The best performing methods, in terms of the rate of false positives on the LabelMe dataset, are 36h15 and ARToolkitPlus-Simple. These two coding families also have the fewest number of distinguishable codes, which gives them an advantage. The other three systems have approximately comparable numbers of distinguishable codes; ARToolkitPlus-BCH performs very poorly; ARTag does much better, and our proposed 36h10 encoding does even better.

ARToolkitPlus's BCH coding scheme has the highest false positive rate, followed by ARTag. Our 36h10 encoding, generated with a minimum complexity of 10, performs better than both of these systems. This is a central result of this paper.
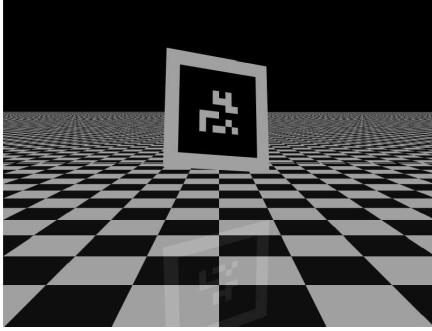

Fig. 8. Example synthetic image. We generated ray-traced images in order to create ground-truthed datasets for our evaluation. In this example, the tag is 10m from the camera, and its normal vector points 30.3 degrees away from the camera.

The plot shows data for two additional schemes: ARTP-Simple performs about the same as our 36h10 encoding, but because its tag family has one quarter as many distinguishable tags, its false positive rate is correspondingly lower. For comparison purposes, we also include the false positive rate for 36h15 family with only 27 distinguishable codewords. As expected, it has an exceptionally low false positive rate.

B. Localization Accuracy

To evaluate the localization accuracy of the detector, we used a ray tracer to generate images with known ground truth (see Fig. 8 for an example). The true location and orientation of the tags was varied randomly and compared to the detected position. Images were generated at a resolution of 400×400 with a pinhole lens and focal length of 400 pixels.

The main factor in localization accuracy is the size of the target, which is affected by both the distance and orientation of the tag. To decouple these factors, we conducted two experiments. The first experiment measured the orientation accuracy of targets while fixing the distance. The critical parameter is the angle ϕ between the target's normal vector
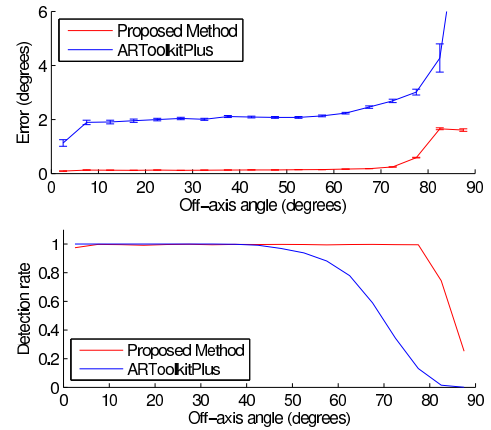

Fig. 9. Orientation accuracy. Using our ray-tracing simulator (which provides ground truth), we evaluated the accuracy of two tag detector's localization accuracy. We fixed the range to the tag and varied the angle between the tag's normal and the camera direction. For both systems, the performance in localization accuracy and in success rate worsens as the tag rotates away from the camera. However, the proposed system has dramatically lower localization error and is able to detect targets more reliably.

and the vector to the camera. When ϕ is 0, the target is facing directly towards the target; as ϕ approaches $\pi/2$, the target rotates out of view and we expect performance to decrease. We measure performance in terms of both the localization accuracy and detection rate. In Fig. 9, we see that our detector significantly outperforms the ARToolkitPlus detector: not only are the orientation and distance estimates more accurate, but it can detect tags over a larger range of ϕ.

The complementary experiment is to hold $\phi = 0$ and to vary the distance. We expect that as distance increases, accuracy will decrease. In Fig. 10, we see that our detector works reliably to 50 m, while the ARToolkitPlus detector's detection rate drops to under 50% at around 25 m. In addition, our detector provides significantly more accurate localization results.

Naturally, real-world performance of the system will be lower than these synthetic experiments due to noise, lighting variation, and other non-idealities (such as lens distortion or tag non-planarity). Still, the real-world performance of our system has been very good.

While our methods are generally more computationally expensive than those used by ARToolkitPlus, our Java implementation runs at interactive rates (30 fps) on VGA resolution images (Intel Core2 CPU at 2.6GHz). Higher resolutions signficantly impact runtime due to the graph-based clustering. We expect significant speedups by making use of SIMD optimizations and accelerated image procesing libraries in our ongoing C port.

VII. CONCLUSION

We have described a visual fiducial system that significantly improves upon previous methods. We described a new approach for detecting edges using a graph-based clustering method along with a coding system that is demonstrably
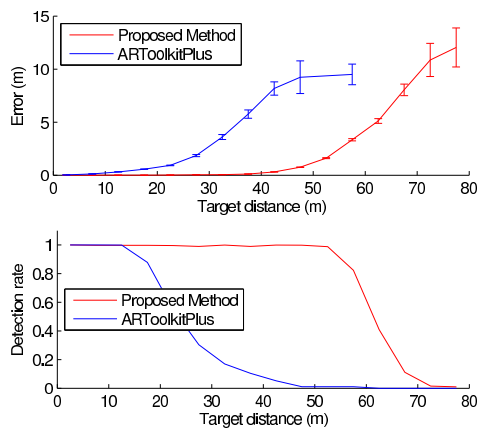
Fig. 10. Distance accuracy. In order to evaluate the accuracy of distance estimation to the target, we again used ground-truthed simulation data. For this experiment, we fixed the target so that it faced the camera but varied the distance between the tag and the camera. Our proposed method significantly outperforms ARToolkitPlus, providing more accurate range estimates, and over twice the working detection range.

stronger than previous methods. We have also described a set of benchmarks that we hope will make it easier to evaluate other methods in the future. In contrast to other systems (with the notable exception of ARToolkit), our implementation is fully open. Our source code and benchmarking software are freely available:

http://april.eecs.umich.edu/

REFERENCES

[1] C.-H. Chu, D.-N. Yang, and M.-S. Chen, "Image stablization for 2d barcode in handheld devices," in *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*. New York, NY, USA: ACM, 2007, pp. 697–706.

[2] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose tracking from natural features on mobile phones," in *IS-MAR '08: Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 125–134.

[3] M. Fiala, "ARTag, a fiducial marker system using digital techniques," in *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 590–596.

[4] A. C. Sementille, L. E. Lourenço, J. R. F. Brega, and I. Rodello, "A motion capture system using passive markers," in *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*. New York, NY, USA: ACM, 2004, pp. 440–447.

[5] J. Sattar, P. Giguère, and G. Dudek, "Sensor-based behavior control for an autonomous underwater vehicle," *Int. J. Rob. Res.*, vol. 28, no. 6, pp. 701–713, 2009.

[6] M. Fiala, "A robot control and augmented reality interface for multiple robots," in *CRV '09: Proceedings of the 2009 Canadian Conference on Computer and Robot Vision*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 31–36.

[7] ——, "Vision guided control of multiple robots," *Computer and Robot Vision, Canadian Conference*, vol. 0, pp. 241–246, 2004.

[8] U. Frese, "Deutsches zentrum für luft- und raumfahrt (DLR) dataset," 2003.

[9] J. Sattar, E. Bourque, P. Giguere, and G. Dudek, "Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction," *Computer and Robot Vision, Canadian Conference*, vol. 0, pp. 165–174, 2007.

[10] T. Lochmatter, P. Roduit, C. Cianci, N. Correll, J. Jacot, and A. Martinoli, "SwisTrack - A Flexible Open Source Tracking Software for Multi-Agent Systems," in *Proceedings of the IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems (IROS 2008)*. IEEE, 2008, pp. 4004–4010. [Online]. Available: http://iros2008.inria.fr/

[11] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *IWAR '99: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*. Washington, DC, USA: IEEE Computer Society, 1999, p. 85.

[12] D. Wagner and D. Schmalstieg, "Making augmented reality practical on mobile phones, part 1," *IEEE Computer Graphics and Applications*, vol. 29, pp. 12–15, 2009.

[13] S.-w. Lee, D.-c. Kim, D.-y. Kim, and T.-d. Han, "Tag detection algorithm for improving the instability problem of an augmented reality," in *ISMAR '06: Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 257–258.

[14] D. Parikh and G. Jancke, "Localization and segmentation of a 2d high capacity color barcode," in *WACV '08: Proceedings of the 2008 IEEE Workshop on Applications of Computer Vision*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 1–6.

[15] P. C. Santos, A. Stork, A. Buaes, C. E. Pereira, and J. Jorge, "A real-time low-cost marker-based multiple camera tracking solution for virtual reality applications," January 2009.

[16] A. Mohan, G. Woo, S. Hiura, Q. Smithwick, and R. Raskar, "Bokode: imperceptible visual tags for camera based interaction from a distance," *ACM Trans. Graph.*, vol. 28, pp. 98:1–98:8, July 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1531326.1531404

[17] J. McLurkin, "Analysis and implementation of distributed algorithms for Multi-Robot systems," Ph.D. thesis, Massachusetts Institute of Technology, 2008.

[18] Y. Yamamoto, P. Pirjanian, M. Munich, E. Dibernardo, L. Goncalves, J. Ostrowski, and N. Karlsson, "Optical sensing for robot perception and localization," in *2005 IEEE Workshop on Advanced Robotics and its Social Impacts*, 2005, pp. 14–17.

[19] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[20] R. L. Rivest and C. E. Leiserson, *Introduction to Algorithms*. New York, NY, USA: McGraw-Hill, Inc., 1990.

[21] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, November 2004.

[22] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[23] K. Shoemake and T. Duff, "Matrix animation and polar decomposition," in *In Proceedings of the conference on Graphics interface 92*. Morgan Kaufmann Publishers Inc, 1992, pp. 258–264.

[24] A. Trachtenberg, A. T. M. S, E. Vardy, and C. L. Liu, "Computational methods in coding theory," Tech. Rep., 1996.

[25] R. A. Brualdi and V. S. Pless, "Greedy codes," *J. Comb. Theory Ser. A*, vol. 64, no. 1, pp. 10–30, 1993.

[26] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: A database and web-based tool for image annotation," Tech. Rep. MIT-CSAIL-TR-2005-056, Massachusetts Institute of Technology, Tech. Rep., 2005.